A constant-round resettably-sound resettable zero-knowledge argument in the BPK model

Seiko Arita

2011.07.28
Graduate School of Information Security,
Institute of Information Security, Japan

Abstract. *In resetting attacks against a proof system, a prover or a verifier is reset and enforced to use the same random tape on various inputs as many times as an adversary may want. Recent deployment of cloud computing gives these attacks a new importance. This paper shows that argument systems for any NP language that are both resettably-sound and resettable zero-knowledge are possible by a constant-round protocol in the BPK model. For that sake, we define and construct a resettably-extractable* conditional *commitment scheme.*

keywords: resettable zero-knowledge, resettable soundness, conditional commitment, resettably extractable.

1 Introduction

Resettable zero-knowledge proofs. The notion of resettable zero-knowledge of proof systems was proposed by Canetti, Goldreich, Goldwasser and Micali [4]. It requires that proofs be zero-knowledge even if a prover is reset and enforced to use the same random tape on various inputs as many times as an adversarial verifier may want. This resetting attack is motivated by attacks against smart cards, where a stolen card can be reset as many times as an attacker wants. Recent deployment of cloud computing gives the notion new importance, because virtual machines in a cloud are far more easier for adversaries to reset than real machines in the user's perimeter [10], [13], [14].

[4] shows that assuming the existence of 2-round perfectly-hiding commitments, there exist resettable zero-knowledge proofs with a polynomial number of rounds for all NP languages. Later, the round complexity is improved to poly-logarithmic number by Kilian and Petrank [11].

[4] also presented the notion of the bare public key (BPK) model. The only requirement in the BPK model is that all verifiers deposit some public-key in a public file before the prover begins the interaction. They constructed constant-round resettable zero-knowledge arguments in the BPK model, assuming some sub-exponential hardness assumption.

Resettably-soundness. Barak, Goldreich, Goldwasser and Lindell [3] proposed the notion of resettably-soundness, the dual notion of resettable zero-knowledge. It requires that proofs be sound even if a verifier is reset and enforced to use the same random tape on various inputs as many times as an adversarial prover may want. Resettably-sound zero-knowledge *proofs* exist only for languages in non-uniform P. Even for arguments no resettably-sound zero-knowledge arguments exist for language outside BPP if the simulator is black-box.

[3] constructed a constant-round resettably-sound zero-knowledge argument for NP only assuming collision-resistant hash functions. In the construction, the Barak's non-black-box constant-round zero-knowledge argument [1], especially the fact that it has only constant-round, plays the essential role.

By utilizing that resettably-sound zero-knowledge argument, [3] also constructed a constant-round sequentially-sound resettable zero-knowledge argument in the BPK model. (We note that soundness in the BPK model is divided into four subcategories; stand-alone, sequential, concurrent and resettable soundness [12].)

Resettably-sound resettable zero-knowledge argument. How about simultaneous resettability ? Crescenzo, Persiano and Visconti [5] showed that a constant-round concurrent-sound resettable zero-knowledge argument is possible in the BPK model assuming some sub-exponential hardness assumption. Moreover, Deng and Lin [8] constructed a constant-round concurrent-sound resettable ZK argument in the BPK model only assuming collision resistant (against polynomial-time adversaries) hash functions.

In the plain model, Deng and Lin [7] showed that if there exist public-coin concurrent zero knowledge arguments for NP, then there exist resettably-sound resettable zero knowledge arguments for NP. However, it is not known whether there exist public-coin concurrent zero knowledge arguments for NP.

Recently, Deng, Goyal and Sahai [6] gave a breakthrough. They gave a new non-back-box strategy based on the Barak's simulator and showed that there exists a both resettably-sound and resettable zero-knowledge argument system for all languages in NP in the plain model, assuming trapdoor permutations and collision-resistant hash function families. Its round-complexity is square in the security parameter.

Our contribution. This paper gives the first constant-round argument system for all NP languages in the BPK model that is both resettably-sound and resettable zero-knowledge.

First, we establish the resettably-soundness, that is, we construct a constant-round resettably-sound concurrent zero-knowledge argument system in the BPK model. To do that, basically we follow the design principle of Deng and Lin [8], which is a variant of Feige and Shamir construction of zero-knowledge argument. However, in the course, we define and construct a new variant of commitment scheme, *resettably-extractable conditional commitment scheme*. In the commitment scheme, simulator can extract the adversary's commitment value even if receiving reset operations at arbitrary moments under some "condition". Thus, by using the resettably-extractable conditional commitment scheme in the Deng and Lin type construction, a verifier-simulator can extract a value committed to by a cheating committer (played by a prover) *even if receiving reset operations at arbitrary moments* under some "condition", and this leads to the resettably-soundness. We will see depending on such "condition" can be justified and obtain a constant-round resettably-sound *concurrent* zero-knowledge argument. Then, we enhance the argument to be resettable zero-knowledge too, by applying to that argument system the transformation of [6], from resettably-sound relaxed-concurrent zero-knowledge arguments to resettably-sound resettable zero-knowledge arguments. That results in a constant-round resettably-sound and resettable zero-knowledge argument system for all NP languages in the BPK model.

2 Definitions

First, we recall some definitions regarding security against resetting attacks, following [4], [3].

Let (P, V) be an interactive proof or argument system for an NP-language L. We denote the possible number of sessions between P and V as $t = \mathrm{poly}(n)$. Let $[t]$ denote a range $\{1, 2, \ldots, t\}$.

We denote statements to be proved as $x = (x_1, \ldots, x_t)$ with their corresponding witnesses $y = (y_1, \ldots, y_t)$ (each y_i witnesses $x_i \in L$).

2.1 Resetting Attack by Cheating Verifiers

A resetting attack by a cheating verifier V^* goes as follows.

1. Choose t independent random tapes $\omega_1, \ldots, \omega_t$. For $i \in [t]$ and $j \in [t]$, let $P_{(i,j)} := P(x_i, y_i; \omega_j)$ denote a prover on input a statement x_i, its witness y_i and a random tape ω_j. We call each $P_{(i,j)}$ an *incarnation* of prover P.
2. A cheating verifier V^* performs a polynomial number of sessions with the incarnations $P_{(i,j)}$ in a sequential manner. Here, V^* can adaptively select which incarnation $P_{(i,j)}$ it has a session with. Same incarnation $P_{(i,j)}$ can be selected many times.
3. V^* halts with an output based on its view. We denote the output as $(P(x, y), V^*(x))$.

Definition 1 (resettable WI) An interactive proof or argument system (P, V) for an NP language L is said to be *resettable witness indistinguishable*, if the following two ensemble are computationally indistinguishable in all resetting attacks by arbitrary feasible cheating verifier V^* :

- $\{(P(x, y^1), V^*(x))\}_{(x,y^1,y^2)}$,
- $\{(P(x, y^2), V^*(x))\}_{(x,y^1,y^2)}$.

Here, both $y^1 = (y_i^1)$ and $y^2 = (y_i^2)$ are sets of witnesses for $x = (x_i)$ to be $x_i \in L$.

Definition 2 (resettable ZK) An interactive proof or argument system (P, V) for an NP language L is said to be *resettable zero-knowledge*, if for any feasible cheating verifier V^* in a resetting attack there exists a feasible simulator M and the following two ensemble are computationally indistinguishable :

- $\{(P(x, y), V^*(x))\}_{(x,y)}$,
- $\{M(x)\}_{(x,y)}$.

Here, $y = (y_i)$ is a set of witnesses for $x = (x_i)$ to be $x_i \in L$.

2.2 Resetting Attack by Cheating Provers

A resetting attack by a cheating prover P^* goes as follows.

1. Choose t independent random tapes $\omega_1, \ldots, \omega_t$. For $j \in [t]$, let $V_{(j)}(x) := V(x; \omega_j)$ denote a verifier on input a statement x and a random tape ω_j. We call each $V_{(j)}(x)$ an *incarnation* of verifier V.
2. A cheating prover P^* performs a polynomial number of sessions with the incarnations $V_{(j)}(x)$ in a sequential manner. P^* can adaptively select a statement x (not necessary in L) to be proved and which incarnation $V_{(j)}$ it has a session with. Same incarnation $V_{(j)}$ can be selected many times on various statements x.

Definition 3 (resettably-sound) An interactive proof or argument system (P, V) for an NP language L is said to be *resettably-sound* if for any feasible cheating prover P^* in a resetting attack, the probability that some incarnation $V_{(j)}(x)$ accepts a false statement $x (\notin L)$ in some session invoked by P^* is negligible.

2.3 The BPK Model

The BPK (bare public key) model is a kind of setup assumption for proof systems proposed by [4], aiming to make it easier to construct a simulator for concurrent or resettable zero-knowledge proof systems.

The only requirement in the BPK model is that all verifiers deposit their public-keys in a public file before any interaction occurs. That is,

- A public file F is a collection of records, each of which is a pair of an index and a public key of a verifier.
- Prover P receives as inputs a statement x, its witness y, a random tape r and the public file F with a verifier's index.
- Verifier V, at the first stage, generates its key pair (pk, sk) and records the public key pk in the public file F. Then, at the second stage, V receives a statement x, its secret key sk and its random tape w as inputs and performs the protocol with P, resulting in an output of accept or reject.

3 Known Constructions

Here, we briefly review what is known about constant-round resettable arguments and constant-round resettably-sound arguments.

3.1 Constant-Round Resettable Arguments

By using the resettably-sound zero-knowledge argument (as seen below), it is shown that there exists a constant-round resettable witness-indistinguishable argument of knowledge, assuming the existence of collision-resistant hash functions (in the plain model) [3].

In the BPK model, there exists a constant-round (concurrent-sound) resettable zero-knowledge argument for all NP languages, assuming collision-resistant hash functions [8].

3.2 Constant-Round Resettably-Sound Arguments

Every constant-round public-coin argument can be converted to a resettably-sound version, preserving prover's property such as witness-indistinguishability and zero-knowledge [3].

It is known that there exists a constant-round public-coin witness-indistinguishable (or zero-knowledge argument) for all NP languages assuming one-way functions (or collision-resistant hash-functions [1]).

Especially, we know that in the plain model:

- there exists a constant-round resettably-sound witness-indistinguishable argument assuming one-way functions, and
- there exists a constant-round resettably-sound zero-knowledge argument assuming collision-resistant hash-functions.

4 Resettably-Extractable Conditional Commitment

Now, we introduce a new variant of commitment scheme, *conditional commitment scheme*. Then we see that *resettably-extractable* conditional commitments are possible by a constant-round protocol in the plain model.

4.1 Conditional commitment

In a *conditional commitment scheme*, a sender takes a *condition* x as input in addition to a string v to be committed. For some given language L, hiding property of conditional commitment is guaranteed only if its condition x is in L, and its binding property is assured only if its condition x is outside from L.

More precisely, it is defined as follows. (In the sequel, notation $(y_1, y_2) \leftarrow (P_1(x_1), P_2(x_2))$ means that by running an interactive protocol (P_1, P_2) on input x_1 for P_1 and x_2 for P_2, parties P_1 and P_2 obtains local outputs y_1 and y_2, respectively.)

Definition 1 (Conditional commitment). *A pair of interactive efficient algorithms (S, R) is said to be* conditional commitment scheme *with respect to a language L if the following three conditions are satisfied :*

- *(Correctness) For any $v \in \{0,1\}^n$ and any $x \in \{0,1\}^n$, it holds that*

$$\Pr[\ (d, c) \leftarrow (S(v, x), R(x)),\ (-, v') \leftarrow (S(d), R(c))\ :\ v' = v\ ]\ =\ 1.$$

- *(Conditional Hiding) For any non-uniform efficient cheating receiver R^* and any $x \in L \cap \{0,1\}^n$, it holds that*

$$\Pr[\ (v_0, v_1, s) \leftarrow R^*(x),\ b \leftarrow \{0,1\},\ (-, b') \leftarrow (S(v_b, x), R^*(s))\ :\ b' = b\ ]\ \leq\ 1/2 + \mathrm{negl}(n).$$

- *(Conditional Binding) For any efficient cheating sender S^* and any $x \in \{0,1\}^n$ that is not in L, it holds that*

$$\Pr[\ (d_1, d_2, c) \leftarrow (S^*(x), R(x)),\ (-, v_1) \leftarrow (S^*(d_1), R(c)),\ (-, v_2) \leftarrow (S^*(d_2), R(c))\ :\ v_1 \neq \perp,\ v_2 \neq \perp,\ v_1 \neq v_2\ ]\ \leq\ \mathrm{negl}(n).$$

4.2 Resettable-Extractability

Our motivation for introducing conditional commitment is in enabling *resettably-extractable* commitment. The resettably-extractable conditional commitment scheme demands that there should be a simulator which can extract a value committed to even by a cheating sender who can mount a resetting attack against a victim receiver, under the constraint that the adversarial commitment is made with respect to the condition x that is not in language L.

First, we define a resetting attack for conditional commitment schemes. For a conditional commitment scheme (S, R), a *resetting attack by a cheating sender S^** goes as follows.

1. Choose t independent random tapes $\omega_1, \ldots, \omega_t$. For $j \in [t]$, let $R_{(j)} := R(\omega_j)$ denote an honest receiver on input random tape ω_j. We call each $R_{(j)}$ an *incarnation* of receiver R.
2. A cheating sender S^* performs a polynomial number of sessions, composed of only commitment phases, with the incarnations $R_{(j)}$ in a sequential manner with respect to some conditions. Here, S^* is supposed to be able to adaptively select a value v and a condition x for commitments and is supposed to be able to choose which incarnation $R_{(j)}$ it has a session with. Same incarnation $R_{(j)}$ can be selected many times on various values v and conditions x.

Definition 4 (Resettable-extractability) A conditional commitment scheme (S, R) with respect to language L is said to be *resettably-extractable* if for every feasible cheating sender S^* mounting a resetting attack against an honest receiver R there exists a feasible *resettable-extractor* E with output

$$(view, value) \leftarrow E(1^n),$$

satisfying that

1. the output *view* is computationally indistinguishable from the real view of S^*, and
2. the output *value* contains *one* of the committed values by S^* with respect to some condition x that is *not* in L in the simulated view *view* of S^* with a noticeable probability $1/\mathrm{poly}(n)$ (if such a terminated commitment with condition outside of L exists).

4.3 Construction of Resettably-Extractable Conditional Commitment Scheme

Our construction of resettably-extractable conditional commitment scheme is based on the commit-with-extract commitment scheme of Barak [2], that is designed to enable a straight-line extractor of strict polynomial time.

We enhances the Barak's commitment scheme so as to be extractable even against a cheating sender who can perform a resetting attack to a victim receiver, as follows.

Protocol 1 – Building Blocks:
- an NP language L,
- a pseudorandom function F,
- a trapdoor permutation f with a hardcore predicate h,
- a (computationally-hiding statistically-binding) non-interactive commitment scheme Com,
- a ZAP (two-round public-coin witness indistinguishable proof [9]) and
- a (stand-alone) constant-round zero-knowledge argument.

– Input for sender S : a condition $x\,(\in \{0,1\}^n)$, a value $v\,(\in \{0,1\}^n)$ and a random tape ω_S.
– Input for receiver R : a condition x and a random tape ω_R.
– The Protocol (Commitment Phase) :
1. $S \to R : S$ selects a trapdoor permutation f with a hardcore predicate h and commits to its random tape $\omega_S : c_S \leftarrow Com(\omega_S)$. S sends f, c_S to R.
2. $S \leftarrow R : R$ computes $\omega_R^* = F_{\omega_R}(x, f, c_S)$. (*From now on to the end of the protocol, R uses ω_R^* as its random tape instead of ω_R.*) R selects a random string r_2 of length n^2 and commits to it : $c_2 \leftarrow Com(r_2)$. R generates a first-round message σ_R of a ZAP. R sends c_2, σ_R to S.
3. $S \to R : S$ selects a random string r_1 of length n^2 and sends to R the string r_1 attached with a ZAP showing that "r_1 is taken from ω_S (that is committed to under c_S), *or* $x \in L$. " If the attached ZAP is invalid, R aborts the protocol immediately.
4. $S \leftarrow R : R$ sends r_2 to S.
5. $S \Leftarrow R : R$ proves to S using the zero-knowledge argument that "there exists a w satisfying $c_2 = Com(r_2; w)$," where
   - every S's message (as a verifier) is coupled with a ZAP claiming that "the sent message is computed using ω_S as a random tape according to the protocol, *or* $x \in L$. " If any of the attached ZAPs is invalid, R aborts the protocol.
6. $S \to R$: (If R's proof is valid) S computes $(r^1, \ldots, r^n) = r = r_1 \oplus r_2$ and takes those inverses $s^i = f^{-1}(r^i)$ for $i = 1$ to n. S computes and sends $C = (v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n))$ to R.

– The Protocol (Decommitment Phase) :
  1. $S \to R : S$ sends $s = (s^1, \ldots, s^n)$ and $v = (v_1, \ldots, v_n)$ to R.
  2. $R : R$ checks whether $r^i = f(s^i)$ for all $i = 1$ to n and whether $C = (v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n))$ or not. If all of the equations hold R outputs v, otherwise outputs $\perp$.

Theorem 1. *Under the assumption that the primitives described in Building Blocks exist, Protocol 1 is resettably-extractable conditional commitment scheme.*

Proof of conditional hiding:
The hiding property of Protocol 1 is inherited from the hiding property of the commit-with-extract commitment scheme of Barak [2]. First we review the Barak's commitment scheme (S', R') (in a slightly changed form (in an unessential way) for our purpose):

– Building Blocks:
  - a trapdoor permutation f with a hardcore predicate h,
  - a (computationally-hiding statistically-binding) non-interactive commitment scheme Com and
  - a (stand-alone) constant-round zero-knowledge argument.
– Input for sender S' : a value $v\,(\in \{0,1\}^n)$.
– Input for receiver R' : a security parameter 1^n.
– The Protocol (Commitment Phase) :
  1. $S' \to R' : S'$ selects a trapdoor permutation f with a hardcore predicate h. S' sends f to R'.
  2. $S' \leftarrow R' : R'$ selects a random string r_2 of length n^2 and commits to it : $c_2 \leftarrow Com(r_2)$. R' sends c_2 to S'.
  3. $S' \to R' : S'$ selects a random string r_1 of length n^2. S' sends r_1 to R'.
  4. $S' \leftarrow R' : R'$ sends r_2 to S'.
  5. $S' \Leftarrow R' : R'$ proves to S' using the zero-knowledge argument that "there exists a w satisfying $c_2 = Com(r_2; w)$."
  6. $S' \to R' : S'$ computes $(r^1, \ldots, r^n) = r = r_1 \oplus r_2$ and takes those inverses $s^i = f^{-1}(r^i)$ for $i = 1$ to n. S' computes and sends $C = (v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n))$ to R'.
– The Protocol (Decommitment Phase) :
  1. $S' \to R' : S'$ sends $s = (s^1, \ldots, s^n)$ and $v = (v_1, \ldots, v_n)$ to R'.
  2. $R' : R'$ checks whether $r^i = f(s^i)$ for all $i = 1$ to n and whether $C = (v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n))$ or not. If all of the equations hold R' outputs v, otherwise outputs $\perp$.

Suppose some efficient cheating receiver R^* breaks the hiding property of Protocol 1 on some condition $x \in L$. We construct a non-uniform efficient cheating receiver $(R')^*$ that breaks the hiding property of the Barak's commitment scheme (S', R') as follows.

Cheating receiver $(R')^*$ on auxiliary input x and w (w is a witness to $x \in L$) works as follows :

– Invoke a copy of R^* and receive its challenge v_0, v_1. Send v_0, v_1 as its own challenge to its challenger S'.
– Receiving f from S', commit to a dummy string (of sufficient length) : $c_S \leftarrow Com(0^*)$. Send c_S, f to the internal R^*.
– Receiving c_2, σ_R from R^*, forward c_2 to S'.

– Receiving r_1 from S', forward it to R^*, appending to it the corresponding ZAP computed from the witness w (instead of the random tape of S').
– Receiving r_2 from R^*, forward it to S'. Then, relay the following zero-knowledge argument between R^* and S'. In the relay of argument, append to every message from S' (to R^*) the corresponding ZAP computed from the witness w (instead of the random tape of S').
– Receiving the final commitment C from S', forward it to R^*.
– When R^* halts, halt with an output that R^* output.

By the hiding property of Com and the witness-indistinguishability of ZAP, it is easy to see that the simulated view of R^* is indistinguishable from its real view. So, the advantage of R^* against S (w.r.t. x) to break the hiding property is essentially the same as the one of $(R')^*$ against S'. But this must be negligible by the hiding property of (S', R').

Proof of conditional binding:
The scheme uses the Blum commitment for the final commitment C and clearly it is "unconditionally" (that is, even if $x \in L$ or not) statistically binding.

Proof sketch of resettable-extractability:
For any feasible cheating sender S^* with respect to conditions $x \notin L$ in a resetting attack, we construct an efficient resettable-extractor E as follows.

Resettable-Extractor E on a security parameter 1^n works as follows:

– Choose t independent random tapes $\omega_R^1, \ldots, \omega_R^t$. For $j \in [t]$, let $R^{(j)} := R(\omega_R^j)$
– Select a random index α that indicates a pair of some incarnation $R^{(j)}$ and its some first message $fmsg$.
– Invoke a copy of S^*.
– For each incarnation $R^{(j)}$, receiving the first message $fmsg = (x, f, c_S)$ from S^*, simulate $R^{(j)}$ as follows:
  • If the current $(R^{(j)}, fmsg)$ does not correspond to the selected index α, simulate the behavior of $R^{(j)}$ completely honestly as in a real run.
  • If the current $(R^{(j)}, fmsg)$ corresponds to the selected index α, do as follows.
    ∗ Check whether $fmsg$ is fresh with respect to j or not. If it is not fresh, retrieve the transcript of the past session between S^* and $R^{(j)}$ that shares the same first message $fmsg$ and simulate this session by taking-and-replaying or by resuming-and-continuing the corresponding messages from the transcript. In the following, we assume $fmsg$ is fresh.
    ∗ Generate a new random tape ω^*. Use ω^* as a random tape for the following simulation of $R^{(j)}$ for this interaction. Commit to a dummy string $0^{n^2} : c_2 \leftarrow Com(0^{n^2})$. Send c_2, σ_R to S^*.
    ∗ Receiving r_1 with a valid ZAP (abort this interaction if any of the ZAPs is invalid), generate a random string $s = s^1, \ldots, s^n$ of length n^2. Compute $r^i = f(s^i)$ for $i = 1$ to n and send $r_2 = r \oplus r_1$ to S^* with $r = r^1, \ldots, r^n$. Then "prove" that the already-sent c_2 is a commitment to r_2 using the zero-knowledge simulator, while checking the ZAPs attached to each S^*'s message. (Abort this interaction if any of the ZAPs is invalid.)
    ∗ Receiving a final commitment C, compute $v_i = C_i \oplus h(s^i)$ for $i = 1, \ldots, n$, and set $value = (v_1, \ldots, v_n)$.

– When S^* halts, output its simulated view and *value*.

Only difference between E's simulation and a real run is in the interaction indexed by $\alpha = (R_{(j)}, fmsg)$.

When the first message $fmsg = (x, f, c_S)$ is not fresh with respect to j, in the real sessions, the random tape $\omega_R^* = F_{\omega_R^j}(fmsg)$ used by $R^{(j)}$ is the same as the one used in the past session between S^* and $R^{(j)}$ that shares the same first message $fmsg$. So, c_2, r_2 are replays of the ones sent in that session. Here, we see that r_1 sent from S^* also must be a replay. In fact, since $x \notin L$, r_1 must be taken from ω_S that is bound by c_S in $fmsg$ by the soundness of the attached ZAPs. (Note that ZAPs are always resettably-sound.) Similarly, all of the messages in the zero-knowledge argument must be also replays by the effect of the attached ZAPs. (Since $x \notin L$, the messages from S^* must be computed using ω_S that is bound by c_S in $fmsg$ by the soundness of the attached ZAPs.) Thus, we know that the above E's simulation by taking-and-replaying or by resuming-and-continuing the transcript of the corresponding past session is valid in the case of non-fresh $fmsg$.

When the first message $fmsg = (x, f, c_S)$ is fresh with respect to j, the incarnation $R^{(j)}$ uses a fresh random tape $\omega_R^* = F_{\omega_R^j}(fmsg)$ by pseudorandomness of F. Therefore, by the hiding property of Com and the zero-knowledge property of the argument, it is easy to see the above simulation by E is indistinguishable from the real cases. Note that the stand-alone zero-knowledge suffices here.

Thus, the simulated view of S^* produced by E is indistinguishable from the real view of S^*.

Finally, in the interaction indexed by $\alpha = (R_{(j)}, fmsg)$, E knows the inverses $s = (s^1, \ldots, s^n)$ of $r^i = f(s^i)$ and E can trivially extract $value = (v_i = C_i \oplus h(s_i))$ from the final commitment C. □

A detailed proof of the resettable-extractability of Protocol 1 is given in the appendix.

Since the primitives in Building Blocks of Protocol 1 exist if trapdoor permutations (for ZAPs) and collision-resistant hash functions (for constant-round ZKA) exist, we have :

Corollary 1. *If trapdoor permutations and collision-resistant hash functions exist, then a constant-round resettably-extractable conditional commitment scheme exists in the plain model.*

5 A Constant-Round Resettably-Sound Resettable Zero-Knowledge Argument in the BPK Model

Here we construct a constant-round argument system for all NP languages in the BPK model that is both resettably-sound and resettable zero-knowledge.

First, making use of the resettably-extractable commitment scheme of the last section, we construct a constant-round *resettably-sound concurrent zero-knowledge* argument system in the BPK model, following the design principle of Deng and Lin [8]. (They constructed a constant-round *concurrent-sound resettable zero-knowledge* argument system.)

Then, we apply to the constructed argument system the transformation of [6] that transforms resettably-sound relaxed-concurrent zero-knowledge arguments into resettably-sound resettable zero-knowledge arguments. That results in a constant-round resettably-sound resettable zero-knowledge argument system for any NP languages in the BPK model.

5.1 Construction of Constant-Round Resettably-Sound Concurrent Zero-Knowledge Argument in the BPK Model

Our resettably-sound concurrent zero-knowledge argument follows the design principle of Deng and Lin [8]. A rough sketch of their principle follows. First, a verifier proves to a prover that it knows its own secret key (or a dummy secret) in a witness-indistinguishable way. Then, the prover commits to a dummy value for the verifier: $c \leftarrow Com(0^n)$. Finally, the prover proves that it knows the witness to the claimed statement or the last commitment c was to the verifier's secret key (or the dummy secret) also in a witness-indistinguishable manner.

In the proof of concurrent-soundness of their argument, a knowledge-extractor (played by verifier-simulator) of the second argument plays a crucial role [8]. It extracts some witness from the cheating prover by rewinding strategy. However, in our situation where a cheating prover P^* can mount a resetting attack against a verifier (beyond a concurrent attack), such rewinding strategy does not work: the extractor must rewind P^* while being get itself rewound.

We address the problem by employing the resettably-extractable conditional commitment scheme from the last section in place of the commitment c.

Let L be an NP language. Our argument system for L in the BPK model works as follows.

Protocol 2 – Building Blocks:

- a one-way function f,
- a constant-round resettably-extractable conditional commitment scheme $rCom$ with respect to L (Section 4),
- a constant-round resettable witness-indistinguishable argument of knowledge for language $\{(y_0, y_1) : \exists\alpha,\ y_0 = f(\alpha) \text{ OR } y_1 = f(\alpha)\}$ (Section 3.1),
- a constant-round resettably-sound witness-indistinguishable argument for language $\{(x, c, y_0, y_1) : \exists\beta,\ \beta \text{ is a witness to } x \in L \text{ OR } \beta = (\beta', r),\ c = rCom(x, \beta'; r),\ y_0 = f(\beta') \text{ OR } \beta = (\beta', r),\ c = rCom(x, \beta'; r),\ y_1 = f(\beta')\}$ (Here, $c = rCom(x, \beta'; r)$ means that the commitment c is a commitment to β' with random tape r under condition x).

– Input for prover P : a statement $x\,(\in L)$, its witness w, a public file F and an index i of a verifier $(pk_i = (f, y_0, y_1) \in F)$.
– Input for verifier V : x and a secret key α (satisfying $y_b = f(\alpha)$).
– The protocol :

1. $P \Leftarrow V : V$ proves to P using the resettable witness-indistinguishable argument of knowledge that "there exists α satisfying $y_0 = f(\alpha)$ OR $y_1 = f(\alpha)$".
2. $P \Rightarrow V : P$ commits to 0^n for V under condition x with transcript c using the resettably-extractable conditional commitment scheme $rCom$.
3. $P \Rightarrow V : P$ proves to V using the resettably-sound witness-indistinguishable argument with witness $\beta = w$ that "there exists β satisfying that β is a witness to $x \in L$ OR $\beta = (\beta', r), c = rCom(x, \beta'; r), y_0 = f(\beta')$ OR $\beta = (\beta', r), c = rCom(x, \beta'; r), y_1 = f(\beta')$."

Theorem 2. *Under the assumption that the primitives listed in Building Blocks exist, Protocol 2 is a constant-round resettably-sound concurrent zero-knowledge argument of L in the BPK model.*

Proof. Completeness of the protocol is immediate.

Concurrent zero-knowledge property of the protocol is proved by a standard argument in the BPK model. Let V^* be any feasible cheating verifier in a concurrent attack. We construct an efficient

simulator Sim for V^* as follows.

Simulator Sim :

- Run the first phase of V^* and receive the public file F.
- Run the second phase of V^* :
  - When V^* passes the first argument in some interaction I with public key pk_i, run the stand-alone non-black-box extractor of the resettable witness-indistinguishable argument of knowledge to extract the corresponding knowledge α_i as to pk_i if not yet extracted. (Note that the stand-alone extractor suffices since the target statement is only pk_i that is fixed in the first phase. See [4] for the full formal argument.)
  - When it is time for Sim (as a prover) to make a commitment c for some interaction I, commit to the extracted value α_i under condition x (that is a claimed statement) with transcript c using $rCom$.
  - When it is time for Sim (as a prover) to perform the second argument for some interaction I, run the prover algorithm for the second argument using $\beta = (\alpha_i, r_i)$ as a witness.

Since F is a polynomial size, the number of invocation of the extractor by Sim is also a polynomial. Hence, Sim runs in a polynomial time. By the conditional hiding property of $rCom$ (note that $x \in L$) and the witness indistinguishable property of the second argument, the simulated view of V^* by Sim is easy to see indistinguishable from real view of V^*. (Use a hybrid experiment where c is a commitment to α_i under x but the second argument uses the real witness w.) This completes proof of the concurrent zero-knowledge property.

Now we prove the resettably-soundness of the protocol. Let P^* be a supposed feasible cheating prover in a resetting attack that convinces some verifier incarnation $V(x)$ on some $x \notin L$ with a non-negligible probability. Using P^* we construct an algorithm A that breaks one-wayness of one-way function f. We can suppose that P^* invokes verifier incarnations $V(x)$ only on x's that are not in L. (We can simulate $V(x)$ honestly if $x \in L$ and we can guess randomly which incarnation $V(x)$ is invoked with respect to $x \notin L$.)

Algorithm A: on input $y\,(= f(\beta))$,

- Choose a random $b \leftarrow \{0,1\}$ and set $y_b = y$. (This defines $\beta_b = \beta$ implicitly.)
- Select a random $\beta_{1-b} \leftarrow \{0,1\}^n$ and set $y_{1-b} = f(\beta_{1-b})$.
- Invoke P^* and give a public file $F = (id, y_0, y_1)$ to it.
- For every incarnation $V(x)$ that P^* invokes, do as follows:
  - Run the prover algorithm of the first argument using β_{1-b} as a witness.
  - For P^*'s commitment c under condition $x\,(\notin L)$ with $rCom$, invoke the resettable-extractor E and simulate its receiver using the view output by E.
  - Run the honest verifier algorithm for the second argument.
- Select a random incarnation $V_{j^*}(x^*)$ (with $x^* \notin L$) that has accepted in the above simulation. The committed value β' under commitment c in the interaction with $V_{j^*}(x^*)$ is being extracted as the *value* output by E with a noticeable probability.
- For the first component β'_1 of β' check whether $f(\beta'_1) = y$ or not. If so output β'_1 otherwise output $\perp$ and halt.

It is obvious that A runs in a polynomial time.

Since the coin b is hidden from P^*, the simulated first argument is the same as the real first argument. Since the view output by the resettable extractor E of $rCom$ is indistinguishable from the real view even for a resetting sender, the simulated commitment c is indistinguishable from the real commitment. The second argument is simulated honestly. Hence, the simulated view of P^* by A is indistinguishable from its real view.

We evaluate the output by A. By the contradictive assumption, the incarnation $V_{j^*}(x^*)$ selected by A is being convinced by P^* on $x^* \notin L$ with a non-negligible probability. Since $x^* \notin L$, the resettably-soundness of the second argument means that its witness β' must be the second or third type : β' includes the inverse images β_0 or β_1 of y_0 or y_1, respectively (as its first component β'_1) and means that β_0 or β_1 must be committed to under c with respect to condition x^*. Therefore, the extracted value β' from c by resettable-extractor E must include β_0 or β_1. By the resettable witness indistinguishability of the first argument, we can see that this extracted value is equal to β_b at probability $1/2$ except with a negligible error. Thus, we know that A outputs $\beta_b = \beta$ with a non-negligible probability, contradicting to the one-wayness of f □

The primitives listed in Building Blocks of Protocol 2 exist if trapdoor permutations and collision-resistant hash functions exist. So, we have:

Corollary 2. *Under the assumption that trapdoor permutations and collision-resistant hash functions exist, there exists a constant-round resettably-sound concurrent zero-knowledge argument for any NP language in the BPK model.*

5.2 Transformation to Resettably-Sound Resettable Zero-Knowledge Argument

Deng, Goyal and Sahai showed a transformation that transforms any resettably-sound relaxed-concurrent zero-knowledge argument into a resettably-sound resettable zero-knowledge argument (The relaxed-concurrent zero-knowledge is a property that weakens concurrent zero-knowledge in some way) [6]. The transformation works in the plain model and preserves the property of being constant rounds. (The transformation uses a resettably-sound zero-knowledge argument as a main building block, that is implemented by a constant-round protocol in a plain model.)

Therefore, by applying the Deng-Goyal-Sahai transformation to Protocol 2, we see that:

Corollary 3. *Under the assumption that trapdoor permutations and collision-resistant hash functions exist, there exists a constant-round resettably-sound resettable zero-knowledge argument for any NP language in the BPK model.*

6 Conclusion

This paper has shown that argument systems for any NP language that are both resettably-sound and resettable zero-knowledge are possible by a constant-round protocol in the BPK model. For that sake, we defined and constructed a resettably-extractable conditional commitment scheme.

References

1. Boaz Barak, "How to Go Beyond the Black-Box Simulation Barrier", In 42nd FOCS, pp. 106-115, 2001.

2. Boaz Barak, "Non-Black-Box Techniques in Cryptography", Thesis for the PH.D. Degree, The Weizmann Institute of Science, 2004.
3. B. Barak and O. Goldreich and S. Goldwasser and Y. Lindell, "Resettably-Sound Zero-Knowledge and its Applications," FOCS 2001, IEEE Computer Society, 2001.
4. Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali, "Resettable zero-knowledge," In STOC, pp. 235-244, 2000.
5. Giovanni Di Crescenzo, Giuseppe Persiano, and Ivan Visconti, "Constant-Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model," CRYPTO 2004, LNCS 3152, pp. 237-253, 2004.
6. Yi Deng, Vipul Goyal, Amit Sahai, "Resolving the Simultaneous Resettability Conjecture and a New Non-Black-Box Simulation Strategy," FOCS 09, pp. 251-260, 2009.
7. Yi Deng and Dongdai Lin, "Instance-Dependent Verifiable Random Functions and Their Application to Simultaneous Resettability," EUROCRYPT 2007, LNCS 4515, pp. 148-168, 2007.
8. Yi Deng and Dongdai Lin, "Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model under Standard Assumption," Inscrypt 2007, LNCS 4990, pp. 123-137, 2008.
9. Cynthia Dwork and Moni Naor, "Zaps and their applications," in FOCS, pp. 283-293, 2000.
10. Tal Garfinkel, Mendel Rosenblum, "When virtual is harder than real: security challenges in virtual machine based computing environments," HOTOS'05, Volume 10, 2005.
11. Joe Kilian and Erez Petrank, "Concurrent and Resettable Zero-Knowledge in Poly-logarithmic Rounds," STOC 01, pp. 560-569, 2001.
12. S. Micali and L. Reyzin, "Soundness in the Public-Key Model," EUROCRYPT 2001, LNCS 2139, pp. 542-565, 2001.
13. T. Ristenpart and S. Yilek , "When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography," Proceedings of Network and Distributed Security Symposium – NDSS '10, 2010.
14. Scott Yilek, "Resettable Public-Key Encryption: How to Encrypt on a Virtual Machine", CT-RSA 2010, LNCS 5985, pp. 41-56, 2010.

A Detailed Proof of the Resettable-Extractability of Protocol 1

We prove the resettable-extractability of Protocol 1. We define a sequence of experiments **Real**, **Hyb1**, ... , **Hyb4**, preserving indistinguishability among their outputs. The first experiment **Real** corresponds to a real resetting attack by a cheating sender S^* against Protocol 1 and the final experiment **Hyb4** gives us the claimed resettable-extractor for the protocol. (In the sequel '$\equiv_c$' (or '$\equiv_s$') denotes computational (or statistical) indistinguishability of both hands.)

Real :

- Choose t independent random tapes $\omega_R^1, \ldots, \omega_R^t$. For $j \in [t]$, let $R^{(j)} := R(\omega_R^j)$.
- Invoke a copy of S^*.
- For each incarnation $R^{(j)}$ that S^* invokes, simulate it as follows:
  - Receiving the first message $fmsg = (x, f, c_S)$ from S^*, compute $\omega_R^* = F_{\omega_{R^{(j)}}}(fmsg)$. Use ω_R^* as a random tape for the following simulation of $R^{(j)}$ for this interaction. Take a random string r_2 of length n^2 and commit to it : $c_2 \leftarrow Com(r_2)$. Send c_2, σ_R to S^*.
  - Receiving r_1 with a valid ZAP (abort this interaction if the ZAP is invalid), send r_2 to S^* and prove that c_2 is a commitment to r_2 using the zero-knowledge argument, while checking the ZAPs attached to each S^*'s message. (Abort this interaction if any of the ZAPs is invalid.)
  - Receive $C = v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n)$.
- When S^* halts, output its simulated view.

The following **Hyb1** differs from **Real** only around the treatment of receiver random-tape for some selected interactions.

Hyb1 :

- Choose t independent random tapes $\omega_R^1, \ldots, \omega_R^t$. For $j \in [t]$, let $R^{(j)} := R(\omega_R^j)$.
- Select a random index α that indicates a pair of some incarnation $R^{(j)}$ and its some first message $fmsg$.
- Invoke a copy of S^*.
- For each incarnation $R^{(j)}$ receiving the first message $fmsg = (x, f, c_S)$ from S^*, simulate it as follows:
  - If the current $(R^{(j)}, fmsg)$ does not correspond to the selected index α, simulate the behavior of $R^{(j)}$ honestly as in **Real**.
  - If the current $(R^{(j)}, fmsg)$ corresponds to the selected index α, do as follows.
    - Check whether $fmsg$ is fresh with respect to j or not. If it is not fresh, retrieve the transcript of the past session between S^* and $R^{(j)}$ that shares the first message $fmsg$ and simulate this session by taking-and-replaying or by resuming-and-continuing the corresponding messages from the transcript. In the following, we assume $fmsg$ is fresh.
    - Generate a new random tape ω^*. Use ω^* as a random tape for the following simulation of $R^{(j)}$ for this interaction. Take a random string r_2 of length n^2 and commit to it : $c_2 \leftarrow Com(r_2)$. Send c_2, σ_R to S^*.
    - Receiving r_1 with a valid ZAP (abort this interaction if any of the ZAPs is invalid), send r_2 to S^* and prove that c_2 is a commitment to r_2 using the zero-knowledge argument, while checking the ZAPs attached to each S^*'s message. (Abort this interaction if any of the ZAPs is invalid.)
    - Receive $C = v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n)$.
- When S^* halts, output its simulated view.

Claim.

$$\textbf{Hyb1} \equiv_c \textbf{Real}.$$

Proof. Only difference between **Hyb1** and **Real** is in the simulation of the interaction indexed by $\alpha = (R_{(j)}, fmsg)$.

Suppose $fmsg$ is fresh with respect to j. Then, $\omega_R^* = F_{\omega_R^{(j)}}(fmsg)$ is indistinguishable from a fresh random-tape by pseudorandomness of F in **Real**. So, the simulation of **Hyb1** is indistinguishable from the simulation of **Real**.

Consider the cases where $fmsg$ is not fresh for j. In **Real**, the random tape $\omega_R^* = F_{\omega_R^{(j)}}(fmsg)$ used by $R^{(j)}$ is the same as the one used in the past session between S^* and $R^{(j)}$ that shares the first message $fmsg$. So, c_2, r_2, that are chosen from ω_R^*, are replays of the ones sent in that session.

We see that r_1 sent from S^* must be also a replay. In fact, since $x \notin L$, r_1 must be taken from ω_S that is bound by c_S in $fmsg$ by the soundness of the attached ZAPs. Note that ZAPs are always resettably-sound.

Similarly, all of the messages in the zero-knowledge argument must be also replays. Since $x \notin L$, the messages from S^* must be computed using ω_S that is bound by c_S in $fmsg$ by the soundness of the attached ZAPs.

Thus, also in non-fresh $fmsg$ case we know that the simulation by **Hyb1** (that replays $R^{(j)}$'s messages in the corresponding past transcript) is indistinguishable from **Real**. □

The next **Hyb2** differs from **Hyb1** only in that **Hyb2** uses the zero-knowledge simulator instead of the real argument to prove that c_2 commits to r_2 for the interaction indexed by $\alpha = (R_{(j)}, fmsg)$.

Hyb2 :

- Choose t independent random tapes $\omega_R^1, \ldots, \omega_R^t$. For $j \in [t]$, let $R^{(j)} := R(\omega_R^j)$.
- Select a random index α that indicates a pair of some incarnation $R^{(j)}$ and its some first message $fmsg$.
- Invoke a copy of S^*.
- For each incarnation $R^{(j)}$, receiving the first message $fmsg = (x, f, c_S)$ from S^*, simulate it as follows:
  - If the current $(R^{(j)}, fmsg)$ does not correspond to the selected index α, simulate the behavior of $R^{(j)}$ honestly as in **Real**
  - If the current $(R^{(j)}, fmsg)$ corresponds to the selected index α, do as follows.
    - Check whether $fmsg$ is fresh with respect to j or not. If it is not fresh, retrieve the transcript of the past session between S^* and $R^{(j)}$ that shares the first message $fmsg$ and simulate this session by taking-and-replaying or by resuming-and-continuing the corresponding messages from the transcript. In the following, we assume $fmsg$ is fresh.
    - Generate a new random tape ω^*. Use ω^* as a random tape for the following simulation of $R^{(j)}$ for this interaction. Take a random string r_2 of length n^2 and commit to it: $c_2 \leftarrow Com(r_2)$. Send c_2, σ_R to S^*.
    - Receiving r_1 with a valid ZAP (abort this interaction if any of the ZAPs is invalid), send r_2 to S^* and "prove" that c_2 is a commitment to r_2 using the zero-knowledge simulator, while checking the ZAPs attached to each S^*'s message. (Abort this interaction if any of the ZAPs is invalid.)
    - Receive $C = v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n)$.
- When S^* halts, output its simulated view.

The stand-alone zero-knowledge property shows :

Claim.

$$\textbf{Hyb2} \equiv_c \textbf{Hyb1}.$$

Proof. Suppose towards the contradiction that **Hyb2** and **Hyb1** is distinguishable by some distinguisher D. Using D, we construct a following cheating verifier V^* against the zero-knowledge property of the argument.

$\mathbf{V}^*$:

- Choose t independent random tapes $\omega_R^1, \ldots, \omega_R^t$. For $j \in [t]$, let $R^{(j)} := R(\omega_R^j)$.
- Select a random index α that indicates a pair of some incarnation $R^{(j)}$ and its some first message $fmsg$.
- Invoke a copy of S^*.
- For each incarnation $R^{(j)}$, receiving the first message $fmsg = (x, f, c_S)$ from S^*, simulate it as follows:
  - If the current $(R^{(j)}, fmsg)$ does not correspond to the selected index α, simulate the behavior of $R^{(j)}$ honestly as in **Real** using the tape ω_R^j.

- If the current $(R^{(j)}, fmsg)$ corresponds to the selected index α, do as follows.
  - Check whether $fmsg$ is fresh with respect to j or not. If it is not fresh, retrieve the transcript of the past session between S^* and $R^{(j)}$ that shares the first message $fmsg$ and simulate this session by taking-and-replaying or by resuming-and-continuing the corresponding messages from the transcript. In the following, we assume $fmsg$ is fresh.
  - Take a random string r_2 of length n^2 and commit to it: $c_2 \leftarrow Com(r_2)$. Send c_2, σ_R to S^*.
  - Receiving r_1 with a valid ZAP (abort this interaction if the ZAP is invalid), send a statement that c_2 is a commitment to r_2 towards its challenger (a prover or a simulator) and transfer to S^* the received proof, while checking the ZAPs attached to each S^*'s message. (Abort this interaction if any of the ZAPs is invalid.)
  - Receive $C = v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n)$.

– When S^* halts, invoke the assumed distinguisher D on input S^*'s simulated view and output its output.

In the above, if the proof that V^* receives from its challenger is a real proof (or a simulated proof respectively), then the simulated view of S^* is the same as the one in **Hyb1** (or in **Hyb2**). This implies that the above V^* violates the zero-knowledge property of the argument. □

The next **Hyb3** differs from **Hyb2** only in that **Hyb3** commits to a dummy string 0^{n^2} for c_2 instead of r_2 for the interaction indexed by $\alpha = (R_{(j)}, fmsg)$.

Hyb3 :

– Choose t independent random tapes $\omega_R^1, \ldots, \omega_R^t$. For $j \in [t]$, let $R^{(j)} := R(\omega_R^j)$.
– Select a random index α that indicates a pair of some incarnation $R^{(j)}$ and its some first message $fmsg$.
– Invoke a copy of S^*.
– For each incarnation $R^{(j)}$, receiving the first message $fmsg = (x, f, c_S)$ from S^*, simulate it as follows:
  - If the current $(R^{(j)}, fmsg)$ does not correspond to the selected index α, simulate the behavior of $R^{(j)}$ honestly as in **Real**
  - If the current $(R^{(j)}, fmsg)$ corresponds to the selected index α, do as follows.
    - Check whether $fmsg$ is fresh with respect to j or not. If it is not fresh, retrieve the transcript of the past session between S^* and $R^{(j)}$ that shares the first message $fmsg$ and simulate this session by taking-and-replaying or by resuming-and-continuing the corresponding messages from the transcript. In the following, we assume $fmsg$ is fresh.
    - Generate a new random tape ω^*. Use ω^* as a random tape for the following simulation of $R^{(j)}$ for this interaction. Commit to a dummy string 0^{n^2}: $c_2 \leftarrow Com(0^{n^2})$. Send c_2, σ_R to S^*.
    - Receiving r_1 with a valid ZAP (abort this interaction if any of the ZAPs is invalid), generate and send r_2 of length n^2 to S^* and "prove" that c_2 is a commitment to r_2 using the zero-knowledge simulator, while checking the ZAPs attached to each S^*'s message. (Abort this interaction if any of the ZAPs is invalid.)
    - Receive $C = v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n)$.

– When S^* halts, output its simulated view.

As usual, the hiding property of Com means :

Claim.

$$\textbf{Hyb3} \equiv_c \textbf{Hyb2}.$$

The final **Hyb4** differs from **Hyb3** only in the way of generating the random string r_2 for the interaction indexed by $\alpha = (R_{(j)}, fmsg)$.

Hyb4 :

- Choose t independent random tapes $\omega_R^1, \ldots, \omega_R^t$. For $j \in [t]$, let $R^{(j)} := R(\omega_R^j)$.
- Select a random index α that indicates a pair of some incarnation $R^{(j)}$ and its some first message $fmsg$.
- Invoke a copy of S^*.
- For each incarnation $R^{(j)}$, receiving the first message $fmsg = (x, f, c_S)$ from S^*, simulate it as follows:
  - If the current $(R^{(j)}, fmsg)$ does not correspond to the selected index α, simulate the behavior of $R^{(j)}$ honestly as in **Real**
  - If the current $(R^{(j)}, fmsg)$ corresponds to the selected index α, do as follows.
    - Check whether $fmsg$ is fresh with respect to j or not. If it is not fresh, retrieve the transcript of the past session between S^* and $R^{(j)}$ that shares the first message $fmsg$ and simulate this session by taking-and-replaying or by resuming-and-continuing the corresponding messages from the transcript. In the following, we assume $fmsg$ is fresh.
    - Generate a new random tape ω^*. Use ω^* as a random tape for the following simulation of $R^{(j)}$ for this interaction. Commit to a dummy string 0^{n^2}: $c_2 \leftarrow Com(0^{n^2})$. Send c_2, σ_R to S^*.
    - Receiving r_1 with a valid ZAP (abort this interaction if any of the ZAPs is invalid), generate a random string $s = s^1, \ldots, s^n$ of length n^2. Compute $r^i = f(s^i)$ for $i = 1$ to n and send $r_2 = r \oplus r_1$ to S^* with $r = r^1, \ldots, r^n$. Then "prove" that c_2 is a commitment to r_2 using the zero-knowledge simulator, while checking the ZAPs attached to each S^*'s message. (Abort this interaction if any of the ZAPs is invalid.)
    - Receive $C = v_1 \oplus h(s^1), \ldots, v_n \oplus h(s^n)$.
- When S^* halts, output its simulated view.

Since r_2 distributes uniformly over $\{0,1\}^{n^2}$ also in **Hyb4**, we have

Claim.

$$\textbf{Hyb4} \equiv_s \textbf{Hyb3}.$$

Now it is immediate that the simulation in **Hyb4** gives the claimed resettable-extractor of Protocol 1. In fact, in **Hyb4** we can efficiently extract the values committed under final commitments C using the knowledge of $s^i = f^{-1}(r^i)$ $(i = 1, \ldots, n)$ for the interaction indexed by $\alpha = (R_{(j)}, fmsg)$. That completes the proof.